波士顿房价预测

1. 特征工程

加载 sklearn 中的波士顿房价数据集会得到506个样本，每个样本有13个属性

```
Boston = datasets.load_boston()
# print(dir(Boston))    # ['DESCR', 'data', 'feature_names', 'filename', 'target']
print(Boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None
```

首先判断数据集集中有没有None缺失值：（**df.info() 函数**）

```
X, Y = Boston.data, Boston.target
df = pd.DataFrame(X,columns=Boston.feature_names)
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
```

```
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
None
```

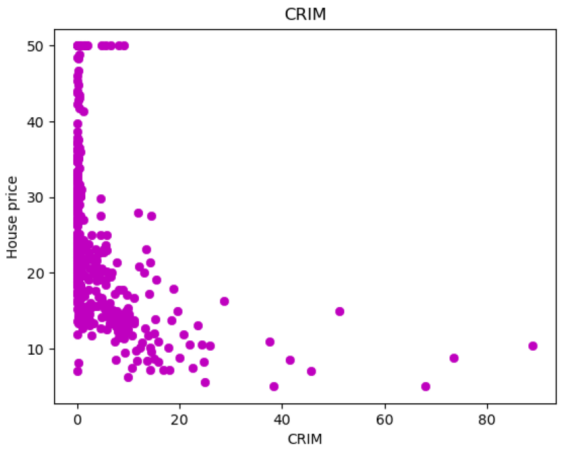
之后可以看一下数据的统计信息：（**df.describe() 函数**）

```
df.describe().to_csv('.\Data\describe.csv')
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
count	506	506	506	506	506	506	506	506	506	506	506	506	506
mean	3.61	11.36	11.14	0.07	0.55	6.28	68.57	3.8	9.55	408.24	18.46	356.67	12.65
std	8.6	23.32	6.86	0.25	0.12	0.7	28.15	2.11	8.71	168.54	2.16	91.29	7.14
min	0.01	0	0.46	0	0.39	3.56	2.9	1.13	1	187	12.6	0.32	1.73
25%	0.08	0	5.19	0	0.45	5.89	45.03	2.1	4	279	17.4	375.38	6.95
50%	0.26	0	9.69	0	0.54	6.21	77.5	3.21	5	330	19.05	391.44	11.36
75%	3.68	12.5	18.1	0	0.62	6.62	94.08	5.19	24	666	20.2	396.23	16.96
max	88.98	100	27.74	1	0.87	8.78	100	12.13	24	711	22	396.9	37.97

观察各属性与房价结果的相关性：

1. **'CRIM'： 人均城镇犯罪率**

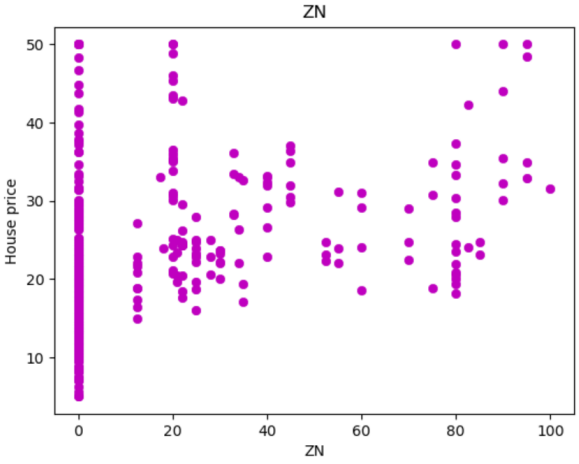


从图上可以清晰地看到，犯罪率越高的城市其房价也相对越低，两者相关度很高。

a. 犯罪率高于 **20%** 的地方房价都在20以下；

b. 犯罪率在 **5%** 以下的地方明显房子售价较高。

2. **'ZN': 占地25,000平方英尺以上的住宅用地比例**



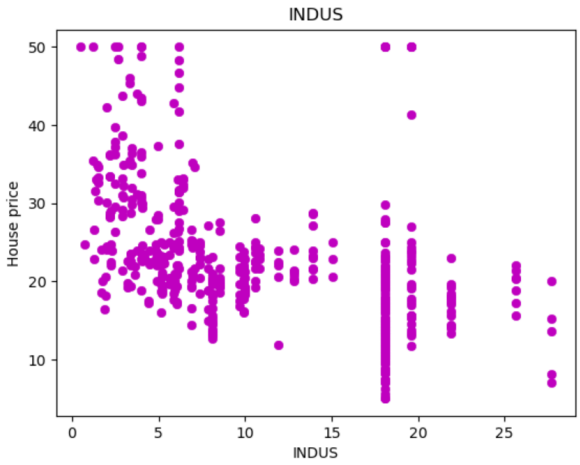
可以看出富人区的起价要高一些，较穷的地方则各档价位都有。

a. ZN在 **0-18**内的区域房价整体偏低；

b. ZN在 **18-40** 内的区域房价整体比较集中；

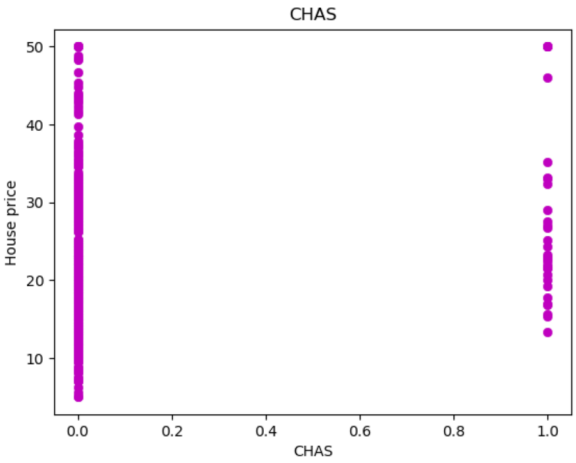
c. ZN大于 **40** 的区域房价整体比较高。

3. **'INDUS': 每个城镇非零售业务用地的比例**

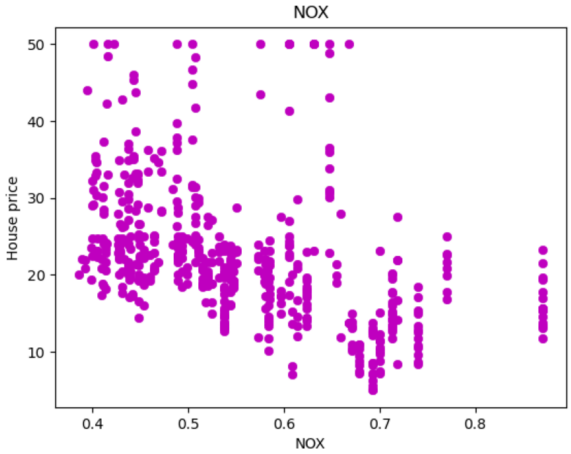


基本上是 INDUS 越低，高价房屋的比例就越高。

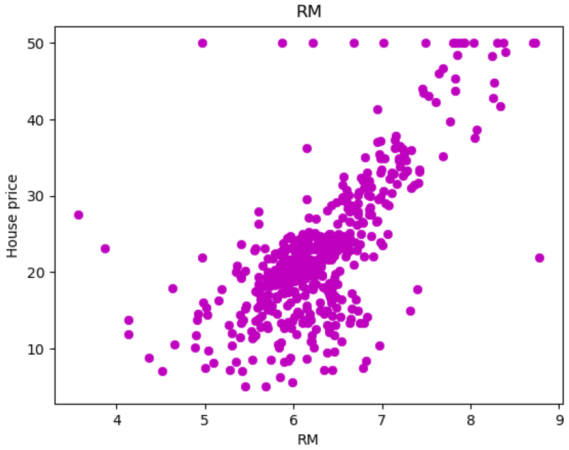
4. **'CHAS': 查尔斯河虚拟变量（如果靠近河流，则为1；否则为0）**



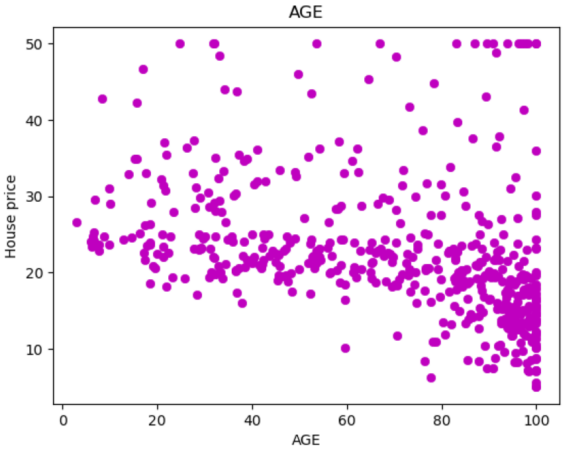
5. **'NOX': 一氧化氮浓度（百万分之一）**



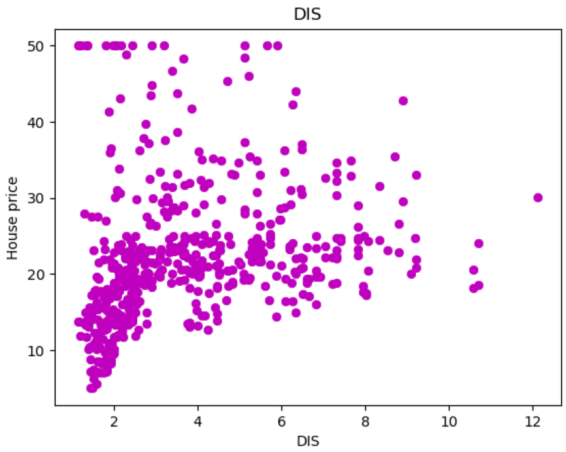
6. **'RM': 每个住宅的平均房间数**



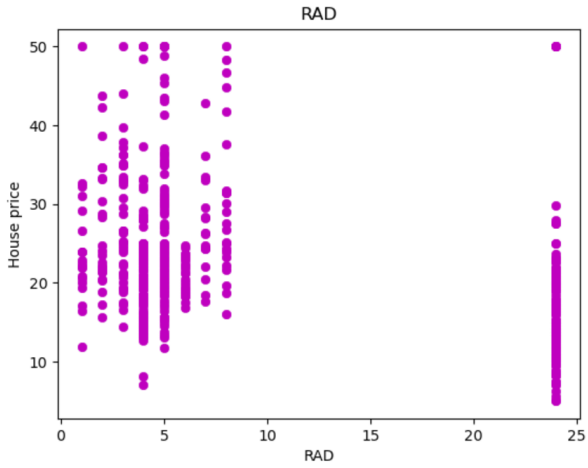
7. **'AGE': 1940年之前建造的自有住房的比例**



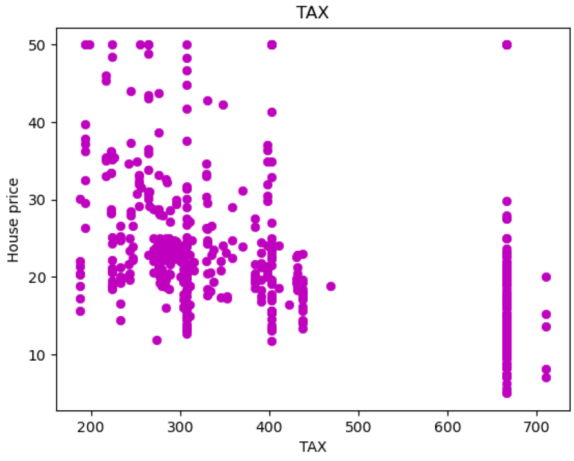
8. **'DIS': 与五个波士顿就业中心的加权距离**



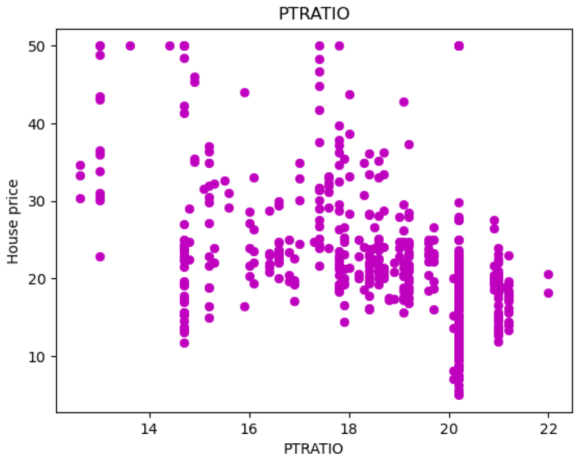
9. **'RAD': 径向公路通达性指数**



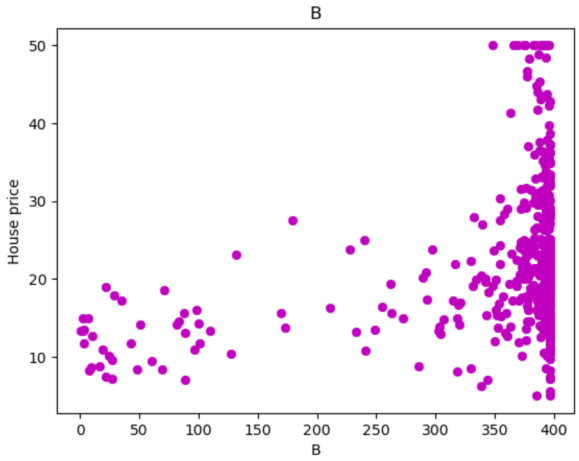
10. **'TAX': 每 10,000美元的全值财产税率**



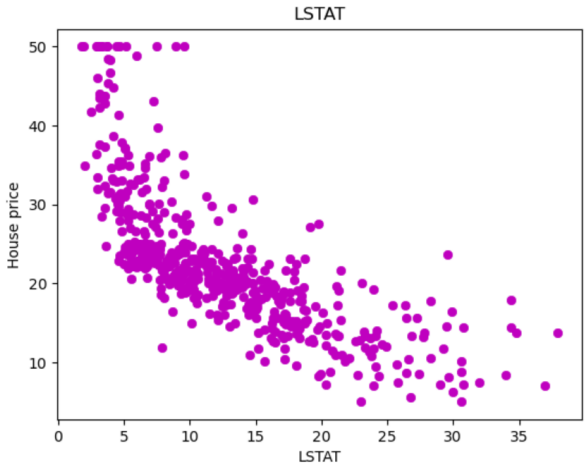
11. **'PTRATIO': 各镇师生比例**



12. **'B': 1000（Bk-0.63）^ 2，其中Bk是按城镇划分的黑人比例**



13. **'LSTAT': 人口地位降低百分比**



归一化 数据集、**清除异常点**（原售价为50的几个点）、**打乱数据集**：

```
X,Y  = preprocessing.scale(X), preprocessing.scale(Y)
df   = pd.DataFrame(X,columns=Boston.feature_names)
df['target'] = Y
df   = df[df.target!=50]
df   = shuffle(df)
```

2. 选择模型

可选的几类回归模型为：

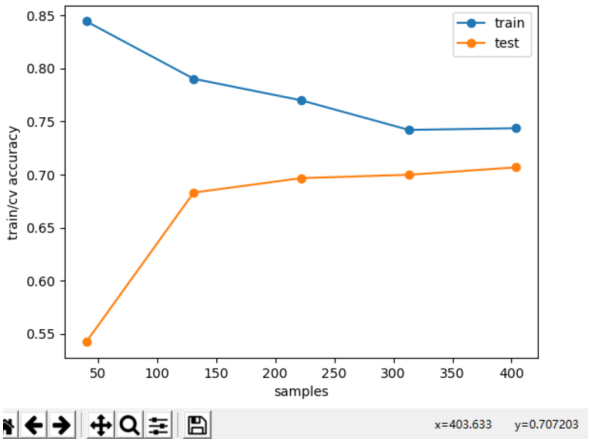
```
from   sklearn import linear_model                          # 线性模型
from   sklearn.neighbors import KNeighborsRegressor         # K近邻模型
from   sklearn.tree      import DecisionTreeRegressor       # 决策树模型
from   sklearn.ensemble  import AdaBoostRegressor           # 集成学习之Adaboost模型
from   sklearn.ensemble  import RandomForestRegressor       # 集成学习之随机森林模型

# model = linear_model.Ridge()
# model = KNeighborsRegressor()
# model = DecisionTreeRegressor()
# model = AdaBoostRegressor()
model = RandomForestRegressor()
```

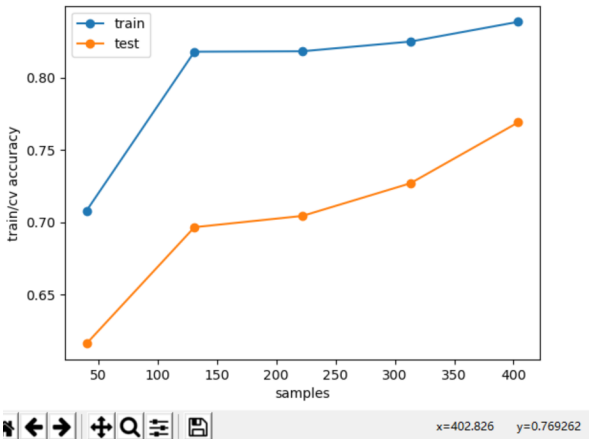
3. 评测模型得分

几种模型的交叉验证得分及学习曲线：

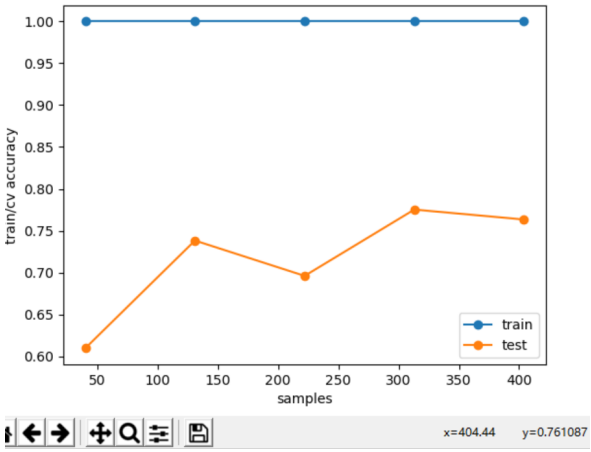
```
linear_model.Ridge()
[0.69418105 0.75820183 0.71820123 0.61048356 0.75312943] 0.7068394188343762
```



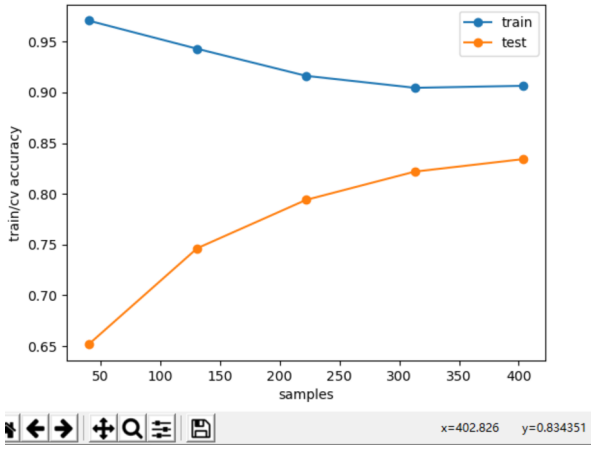

```
KNeighborsRegressor()
[0.74435206 0.87436817 0.76183272 0.76172031 0.69996935] 0.7684485210361188
```



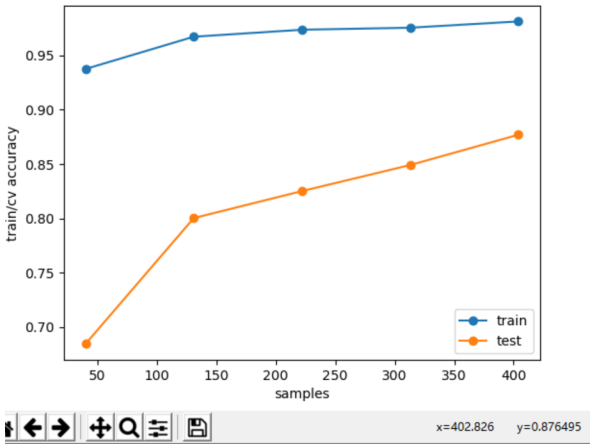

```
DecisionTreeRegressor()
[0.77990121 0.52570365 0.85417234 0.79760376 0.76679541] 0.7448352728639855
```




```
AdaBoostRegressor()
[0.87810629 0.79509021 0.79792583 0.79920706 0.86117552] 0.8263009831599423
```




```
RandomForestRegressor()
[0.81968464 0.89595416 0.85975364 0.91379967 0.89497661] 0.8768337445053292
```



附代码：

```
from   sklearn import datasets
from   sklearn import preprocessing
import pandas  as pd
```

```
import matplotlib.pyplot as plt
from   sklearn import linear_model
from   sklearn.neighbors import KNeighborsRegressor
from   sklearn.tree      import DecisionTreeRegressor
from   sklearn.ensemble  import AdaBoostRegressor
from   sklearn.ensemble  import RandomForestRegressor
from   sklearn.model_selection import train_test_split
from   sklearn.model_selection import cross_val_score
from   sklearn.model_selection import learning_curve
from   sklearn.utils import shuffle
import numpy as np

## ------------------------------------------------------
##                    特征工程
## ------------------------------------------------------
Boston = datasets.load_boston()
# print(dir(Boston))               # ['DESCR', 'data', 'feature_names', 'filename', 'target']
# print(Boston.DESCR)
# print(Boston.feature_names)         # ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO' 'B' 'LSTAT']
# print(Boston.DESCR)
X, Y = Boston.data, Boston.target
X,Y  = preprocessing.scale(X), preprocessing.scale(Y)
df   = pd.DataFrame(X,columns=Boston.feature_names)
df['target'] = Y
df   = df[df.target!=2.99]
df   = shuffle(df)
# df.to_csv('.\Data\data.csv')
# print(df.info())                 # 506 non-null    float64
# df.describe().to_csv('.\Data\describe.csv')
# plt.scatter(df['LSTAT'],Y,c='m',linewidths=0.5)
# plt.title('LSTAT'); plt.xlabel('LSTAT'); plt.ylabel('House price')
# plt.show()
## ------------------------------------------------------
##                    定义模型
## ------------------------------------------------------
# model = linear_model.Ridge()
# model = KNeighborsRegressor()
# model = DecisionTreeRegressor()
# model = AdaBoostRegressor()
model = RandomForestRegressor()
## ------------------------------------------------------
##                    预测/验证
## ------------------------------------------------------
# ['B','PTRATIO','TAX','RAD','NOX','INDUS','ZN','CHAS']
# df = df.drop(['B'],axis=1)
X, Y = df.iloc[:,0:-1], df.iloc[:,-1]
score = cross_val_score(model, X, Y, cv=5)
print(score, np.mean(score))
train_sizes, train_scores, test_scores = learning_curve(model,X,Y,cv=5)
train_scores_mean = np.mean(train_scores,axis=1)
test_scores_mean  = np.mean(test_scores,axis=1)
plt.plot(train_sizes,train_scores_mean,'o-',label='train')
plt.plot(train_sizes,test_scores_mean,'o-',label='test')
plt.xlabel('samples')
plt.ylabel('train/cv accuracy')
plt.legend()
plt.show()
```